

**Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)**

This listing of claims will replace all prior versions, and listings, of claims in the application:

What is claimed is:

1. (Currently amended) In a database system employing a transaction log, an improved method for restoring databases to a consistent version supporting read-only uses, the method comprising:

providing a shared cache storing database blocks in memory of the database system for use by multiple databases;

for in response to a read-only transaction of a given database, creating a cache view of a the given database using the given database's transaction log by logically undoing transactions which have begun but have yet to commit, said cache view comprising particular database blocks in of the shared cache that record a transactionally consistent version of the given view of a particular version of the database at a given point in time supporting read-only uses;

creating a shadow cache for temporarily storing any database blocks that overflow said shared cache in a temporary database view during use of the cache view by the read-only transaction; and

in conjunction with the cache view in the shared cache and the shadow cache, preserving a logical undo operation for the read-only transaction of the given database for logically undoing transactions which have begun but have yet to commit; and

performing the read-only transaction using the cache view and returning results of the read-only transaction, logical undo operation in order to reconstruct a transactionally-consistent prior version of the given database upon starting the read-only transaction, thereby a result comprising a transactionally consistent version of the given database supporting read-only uses.

2. (Previously presented) The method of claim 1, wherein during occurrence of the read-only transaction any database blocks associated with the cache view are not

written from the shared cache to the given database.

3. (Currently amended) The method of claim 1, wherein the blocks that overflow the shared cache are temporarily stored in shadow cache is implemented via a temporary database table.

4. (Canceled)

5. (Currently amended) The method of claim 1, wherein the temporary database shadow cache is used only in the event the cache view overflows the shared cache view.

6. (Currently amended) The method of claim 1, further comprising:
using providing an allocation bitmap for indicating database blocks temporarily stored in use in the temporary database shadow cache.

7. (Currently amended) The method of claim 6, further comprising:
upon completion of the read-only transaction, deleting the blocks temporarily stored in the temporary database shadow cache by updating the allocation bitmap ~~for allocated database blocks.~~

8. (Currently amended) The method of claim 1, wherein said step of temporarily storing database blocks overflowing the shared cache includes storing a mapping to said database blocks in the shadow cache comprises a table in the temporary database table including a first column for maintaining a block number of a cache view block having undo/redo records applied to it and a second column for maintaining a block number allocated to temporarily store said database blocks overflowing the shared cachesave off a modified block from the cache view.

9. (Canceled)

10. (Original) The method of claim 1, further comprising:

upon termination of the read-only transaction, marking the cache view as closed.

11. (Currently amended) The method of claim 10, further comprising:
when new block allocations need to be made in the shared cache, traversing the shared cache looking for database blocks to purge; and
purging database blocks from any cache view that ~~have~~has been marked as closed.

12. (Currently amended) The method of claim 1, further comprising:
~~reusing~~sharing the cache view created for the read-only transaction ~~with~~for other read-only transactions; which start within a specified period of time following the start of the read-only transaction.

13. (Currently amended) The method of claim 1, further comprising:
detecting the read-only transaction; and
upon occurrence of write operations, adding back link log records to the database's transaction log that serve to link together ~~blocks-log records~~ of the transaction log that pertain to a write ~~the read-only~~ transaction.

14. (Currently amended) The method of claim 13, further comprising:
if the read-only transaction must be undone, using the back link log records to skip portions of the transaction log that are irrelevant for undoing an uncommitted write transaction ~~the read-only transaction~~, wherein the back link log records are only generated in the transaction log when there are active read only transactions.

15. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 1.

16. (Currently amended) The method of claim 1, further comprising:
~~A downloadable~~ downloading a set of processor-executable instructions for performing the method of claim 1.

17. (Currently amended) A database system capable of restoring databases to a consistent version supporting read-only uses, the system comprising:

a computer having a processor and memory;

a log manager module which manages a transaction log of the database system;

a cache manager module for managing a shared cache that stores database blocks in memory of the database system for use by multiple databases and creating a cache view of a given database ~~created~~ using the transaction log of the given database, said cache view being created in response to a read-only transaction of the given database, said cache view comprising particular database blocks of the shared cache that record a view of a particular version of the given database at a given point in time, wherein the cache manager utilizes a temporary database shadow-cache for storing any database blocks that overflow said shared cache view during use of the cache view by the read-only transaction and invokes a transaction manager module for logically undoing transactions which have begun but have yet to commit in creating the cache view; and

a transaction manager module for ~~performing read-only transactions of the database system and which performs a logical undo operation for the read-only transaction of the given database for~~ logically undoing transactions which have begun but have yet to commit upon starting the read-only transaction in order to ~~reconstruct the~~ cache view comprising a transactionally consistent prior version of the given database ~~upon starting the read-only transaction, performing the read-only transaction using the cache view, and returning results of the read-only transaction thereby returning a result comprising a transactionally consistent version of the given database supporting read-only uses.~~

18. (Previously presented) The system of claim 17, wherein during occurrence of the read-only transaction any database blocks associated with the cache view are not written from the shared cache to the given database.

19. (Currently amended) The system of claim 17, wherein the cache manager stores database blocks that overflow the shared cache in shadow-cache ~~is implemented~~

via a temporary database table.

20. (Canceled)

21. (Currently amended) The system of claim 17, wherein the temporary database shadow cache is used only in the event the cache view overflows the shared cache view.

22. (Currently amended) The system of claim 17, wherein said cache manager maintains an allocation bitmap indicating database blocks temporarily stored in use in the temporary database shadow cache.

23. (Currently amended) The system of claim 22, wherein said cache manager deletes blocks from the temporary database the shadow cache by updating the allocation bitmap for allocated database blocks.

24. (Currently amended) The system of claim 17, wherein said cache manager stores a mapping to database blocks overflowing the shared cache in a table of the shadow cache comprises a temporary database table including a first column for maintaining a block number of a cache view block having undo/redo records applied to it and a second column for maintaining a block number in a temporary database allocated to temporarily store said database blocks overflowing the shared cache ~~save off a modified block from the cache views~~.

25. (Canceled)

26. (Previously presented) The system of claim 17, wherein said cache manager marks the cache view as closed, upon termination of the read-only transaction.

27. (Currently amended) The system of claim 26, wherein said cache manager traverses the shared cache looking for database blocks to purge, and purges database blocks from any cache view that have~~has~~ been marked as closed when new block

allocations need to be made in the shared cache.

28. (Currently amended) The system of claim 17, wherein said cache manager ~~shares~~ reuses the cache view created for the read-only transaction ~~with~~ for other read-only transactions which start within a specified period of time following the start of the read-only transaction.

29. (Currently amended) The system of claim 17, wherein said log manager detects the read-only transaction, and adds back link log records to the transaction log that serve to link together ~~blocks-log records~~ of the transaction log that pertain to a write transaction that may need to be logically undone~~the read-only transaction.~~

30. (Currently amended) The system of claim 29, wherein said log manager uses the back link log records to skip portions of the transaction log that are irrelevant for undoing the ~~read-only~~write transaction, wherein the back link log records are only generated in the transaction log when there are active read only transactions.